시뮬레이터를 활용한 K-City Map 기반 자율주행 알고리즘 개발

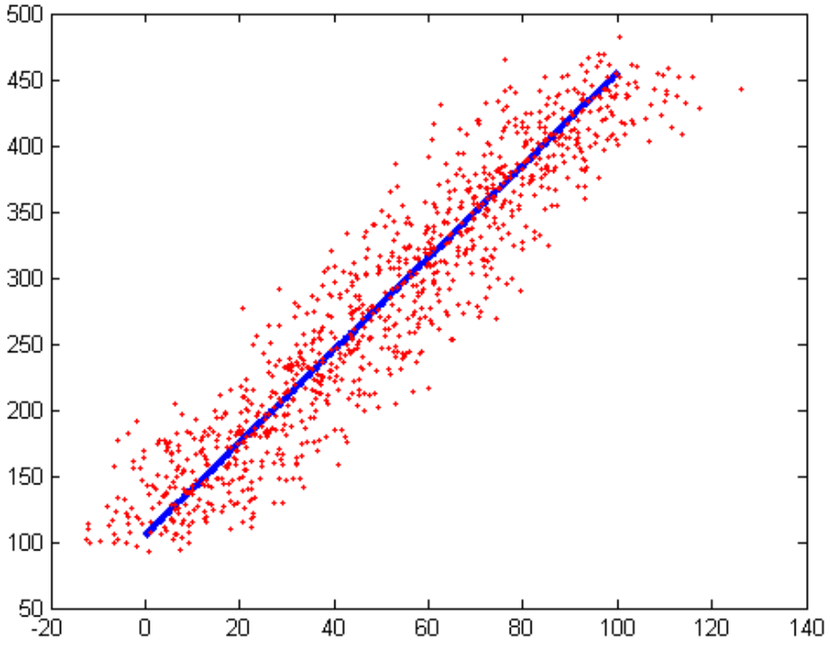
프로젝트 지향 자율주행차 전문인력 양성과정

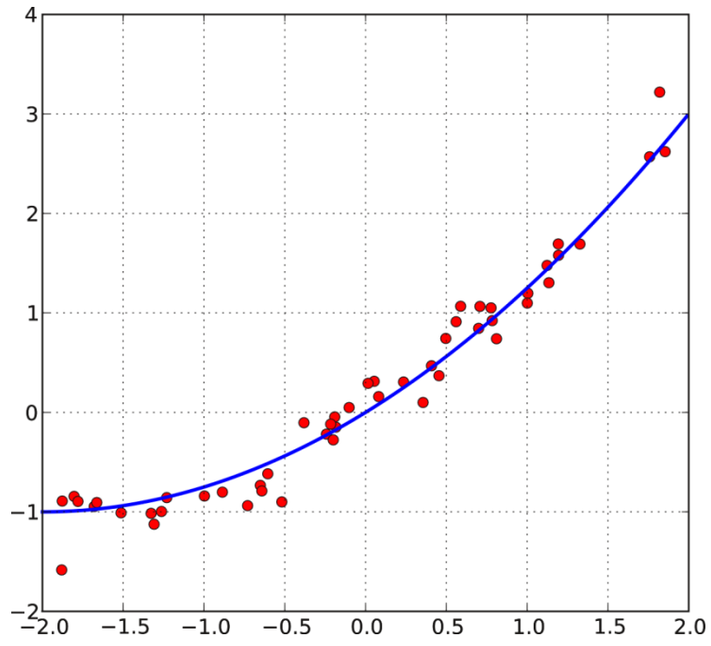
목차

1. 최소 자승법

최소자승법

- 최소 자승법(Least Mean Square)
  - 어떤 점들의 분포를 직선이나 곡선으로 근사화하는 방법
  - 수학적 도구, 수치해석, 회귀분석, 영상처리 다양한 분야에서 사용한다.
  - 어떤 모델의 파라미터를 구하는 한 방법으로서, 데이터와 residual의 합을 최소화하도록 모델의 파라미터를 구하는 방법
  - residual은 어떤 데이터가 추정된 모델로부터 얼마나 떨어진 값인가를 나타내는 용어이다.
  - 대수적 방법, 해석학적 방법, 비선형 최소자승법

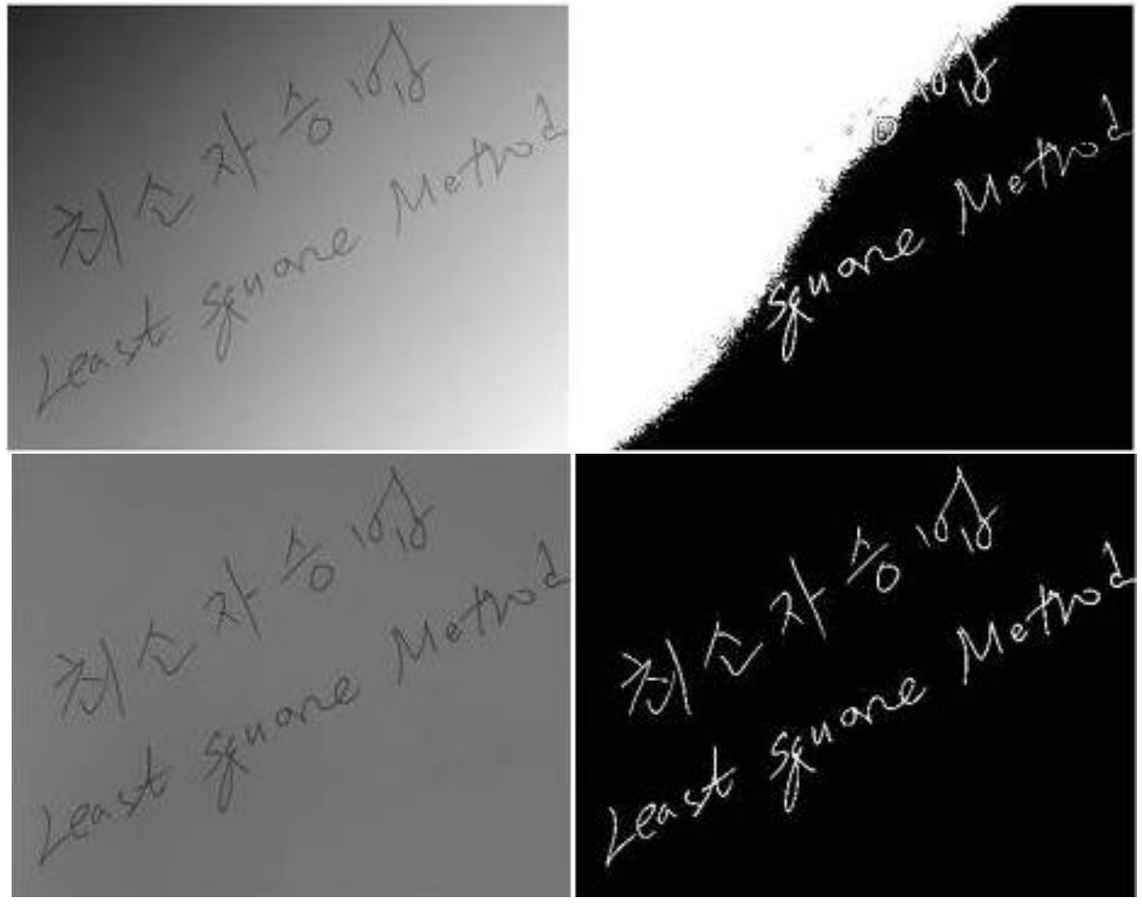




https://darkpgmr.tistory.com/56

최소자승법

- 최소 자승법(Least Mean Square)
  - 어떤 점들의 분포를 직선이나 곡선으로 근사화하는 방법
  - 수학적 도구, 수치해석, 회귀분석, 영상처리 다양한 분야에서 사용한다.
  - 어떤 모델의 파라미터를 구하는 한 방법으로서, 데이터와 residual의 합을 최소화하도록 모델의 파라미터를 구하는 방법
  - residual은 어떤 데이터가 추정된 모델로부터 얼마나 떨어진 값인가를 나타내는 용어이다.
  - 대수적 방법, 해석학적 방법, 비선형 최소자승법



https://darkpgmr.tistory.com/56

최소자승법

- 대수적 방법
  - 행렬식 형태로 표현한 후에 선형대수학을 적용하는 방법으로 모델 파라미터를 구한다.
  - pseudo inverse 라는 방법을 이용해 계산한다.

$$
\begin{aligned}
ax_1+b&=y_1\\
ax_2+b&=y_2\\
&\vdots\\
ax_n+b&=y_n
\end{aligned}
\qquad
\begin{pmatrix} x_1 & 1\\ \vdots & \vdots\\ x_n & 1 \end{pmatrix}\begin{pmatrix} a\\ b \end{pmatrix}=\begin{pmatrix} y_1\\ \vdots\\ y_n \end{pmatrix}
\qquad
\begin{aligned}
X &= pinv(A)B\\
&= (A^TA)^{-1}A^TB
\end{aligned}
$$

$AX=B$

https://darkpgmr.tistory.com/56

최소자승법

- 해석학적 방법
  - 모델 파라미터들로 편미분한 후에 그 결과를 0으로 놓고 연립 방정식을 푸는 방법
  - 다소 복잡함

$$\frac{\partial}{\partial a}\sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} 2(y_i - ax_i - b)(-x_i) = 0$$

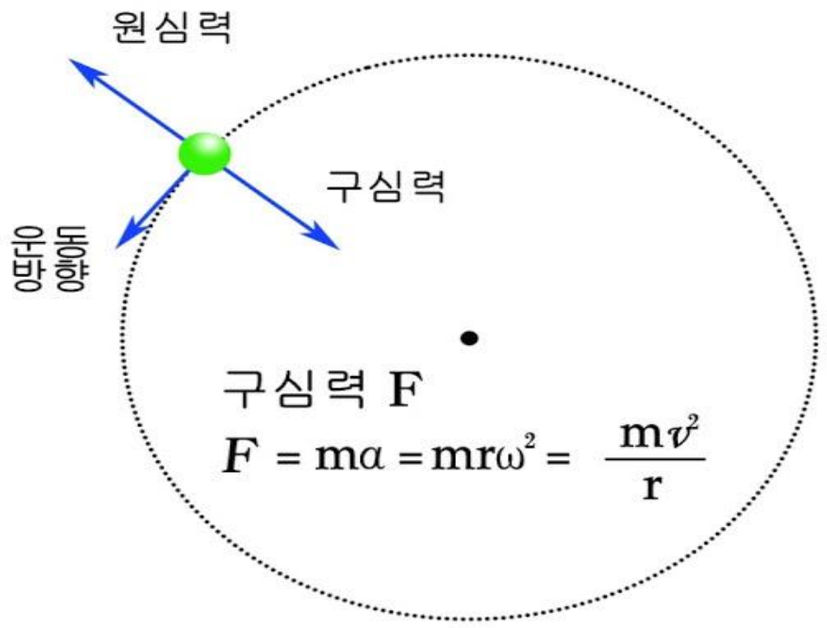
$$\frac{\partial}{\partial b}\sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} 2(y_i - ax_i - b)(-1) = 0$$

https://darkpgmr.tistory.com/56

2. 경로 기반 속도 계획

경로 기반 속도 계획

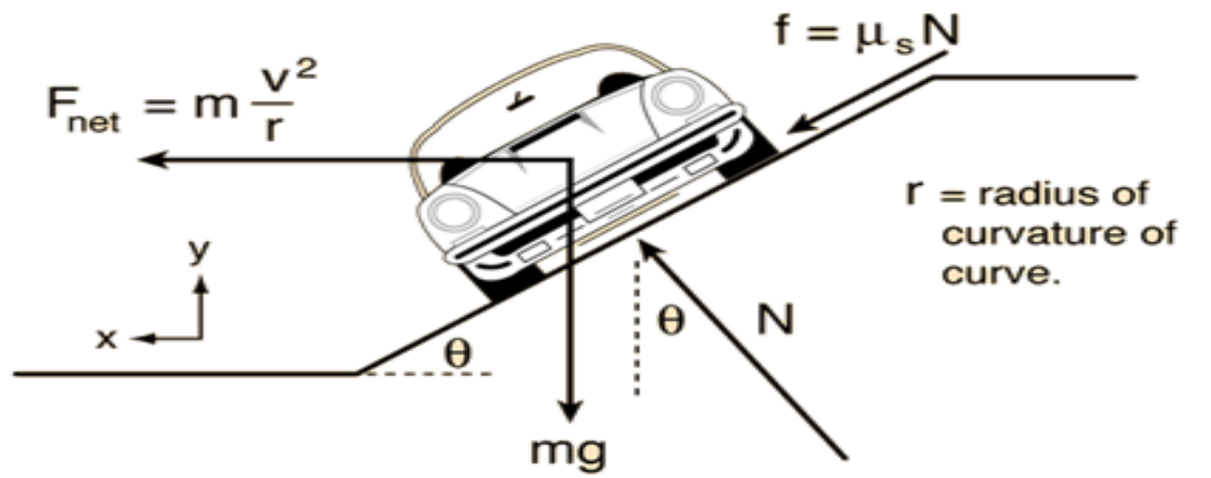
- 경로 기반 속도 계획
  - 경로 기반의 속도 계획은 말 그대로 경로에서 얼마만큼의 속도를 내야 하는가를 계획하는 것
  - 차량은 직선에서 비교적 빠른 속도로 주행이 가능하지만, 곡선에서는 빠른 속도로 주행할 수 없다.
  - 곡선을 주행할 때 구심력의 작용 반작용 힘에 의해 차량은 원심력을 받게된다. 이 힘은 회전반경에 반비례, 속도의 제곱에 비례하기 때문에 속도가 클 수록, 회전방경이 작을수록 차량이 쉽게 전복된다.



https://

경로 기반 속도 계획

- 경로 기반 속도 계획
  - 도로에서 받는 차량의 힘으로 부터 차량이 전복되지 않는 주행가능한 최대속도를 구할 수 있다.
  - r은 경로의 곡률반지름, g는 중력, us는 운동 마찰력



Force equations at maximum speed v, at threshold of sliding up incline.

$$\Sigma F_x = m\frac{v^2}{r} = N \sin\theta + \mu_s N \cos\theta$$

$$\Sigma F_y = 0 = N \cos\theta - \mu_s N \sin\theta - mg$$

Solving this pair of equations for the maximum speed v gives:

$$v_{max} = \sqrt{\frac{rg(\sin\theta + \mu_s \cos\theta)}{\cos\theta - \mu_s \sin\theta}}$$

The limiting cases are:

$$v_{max} = \sqrt{rg \tan\theta}$$

Frictionless case

$$v_{max} = \sqrt{rg\,\mu_s}$$

Flat roadway

경로 기반 속도 계획

- 경로 기반 속도 계획
  - 경로가 주어졌을 때 앞에서 배운 최소자승법을 이용해서 곡률반지름 r을 구할 수 있다.
  - 경로점들을 이용해 원의 방정식으로 근사화

$$(x-a)^2+(y-b)^2=r^2$$
$$x^2+y^2-2ax-2by+a^2+b^2-r^2=0$$
$$c=a^2+b^2-r^2$$
$$x^2+y^2-2ax-2by+c=0\,(a,b,c\text{에 대해서 정리})$$
$$-2ax-2by+c=-x^2-y^2$$

$$\begin{pmatrix} -2x_1 & -2y_1 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{pmatrix}\begin{pmatrix} a \\ b \\ c \end{pmatrix}=\begin{pmatrix} -x_1^2-y_1^2 \\ \vdots \\ -x_n^2-y_n^2 \end{pmatrix}$$

$$AX=B$$
$$X=(A^TA)^{-1}A^TB$$
$$r=\sqrt{a^2+b^2-c}$$

경로 기반 속도 계획

- 실습
  - 해당 클래스는 전역 경로를 받아서 각 경로점에서 차량의 최대속도를 계산한다.
  - 따라서 path_pub 노드에서 클래스를 선언해서 사용해준다.
  - 계산된 속도는 pure pursuit 노드에 전달 되어야 한다.

```python
class velocityPlanning :
    def __init__(self,car_max_speed,road_friction):
        self.car_max_speed=car_max_speed
        self.road_friction=road_friction

    def curveBasedVelocity(self,global_path,point_num):
        out_vel_plan=[]
        for i in range(0,point_num):
            out_vel_plan.append(self.car_max_speed)
        for i in range(point_num,len(global_path.poses)-point_num):
            x_list=[]
            y_list=[]
            for box in  range(-point_num,point_num):
                x=global_path.poses[i+box].pose.position.x
                y=global_path.poses[i+box].pose.position.y
                x_list.append([-2*x,-2*y,1])
                y_list.append(-(x*x)-(y*y))

            x_matrix=np.array(x_list)
            y_matrix=np.array(y_list)
            x_trans=x_matrix.T

            a_matrix=np.linalg.inv(x_trans.dot(x_matrix)).dot(x_trans).dot(y_matrix)
            a=a_matrix[0]
            b=a_matrix[1]
            c=a_matrix[2]
            r=sqrt(a*a+b*b-c)
            v_max=sqrt(r*9.8*self.road_friction)*3.6  #0.7
            if v_max>self.car_max_speed :
                v_max=self.car_max_speed
            out_vel_plan.append(v_max)

        for i in range(len(global_path.poses)-point_num,len(global_path.poses)):
            out_vel_plan.append(self.car_max_speed)
        return out_vel_plan
```

END